MCC-DSM Specifications

MCC Design Group

Output Data Format

Issue:	DRAFT
Revision:	0.1
Reference:	ATLAS ID-xx
Created:	30 September 2002
Last modified:	7 October 2002 03:14
Edited By:	R. Beccherle and G. Darbo ATLAS / Pixel Collaboration

Abstract

This document describes the Output Data Format of the MCC-DSM.

Table Of Contents

List of Figures

List of Tables

1 Output Data Format

This chapter deals with all output data formats provided by the *MCC*.

The *MCC* has to provide communication with 16 *FE* chips hosted on a module and with the ROD both for configuration and Data Taking. Both the communications between *MCC* and the *FE*'s and the *ROD* use a very simple serial command protocol that is explained in the following chapters.

The first two sections deal with the configuration protocol of the *MCC* and the *FE* chips, section 3 explains the event format between the *FE* chips and the *MCC* while section 4 deals with the *MCC* to *ROD* event format.

1.1 ROD to MCC Configuration Data Format

As described in the Chapter 1, "Command Decoder" all configuration data between the *ROD* and the *MCC* follow a simple serial command protocol. This protocol is subdivided in three major command classes, Trigger, Fast and Slow commands.

All *MCC* configuration commands are Slow commands and eventually take the *MCC* out of RunMode.

GlobalResetMCC is the reset command that initializes all default values of the chip and is the first command to be issued in order to start the configuration phase. The reset of the MCC is synchronous and therefore this command only has an effect on the chip if the clock is turned on.

The WrRegister command is used to initialize all internal registers to their correct state. This command does not produce any output at the *MCC* DTO-0 and DTO-1 pins.

There are four configuration commands that are provided for testing purposes. These commands are RdRegister, WrFifo, RdFifo and WrReceiver.

The WrFifo command allows to completely write any of the 16 data FIFO's present in the MCC, while RdRegister and RdFifo allow to check the contents of any register or FIFO respectively. These two commands are the only *MCC* configuration commands that produce an output. Figure 1-1shows an example of a RdRegister command. As can be seen in response to a RdRegister command, after a certain delay the *MCC* presents on its output the 5 bit header (11101) immediately followed by the 16 bits of the contents of the read register.

⚠ All outputs generated by the *MCC* in response to a configuration command are always sent to the *ROD* at a 40 Mbit/s rate on both links.

The last configuration command is EnDataTake which is used to set the *MCC* in RunMode.

Once in RunMode the *MCC* is able to decode Fast and Trigger commands that are used to perform Event building.

For a complete reference on Slow commands please refer to Chapter 1.2.3, "Slow Commands".

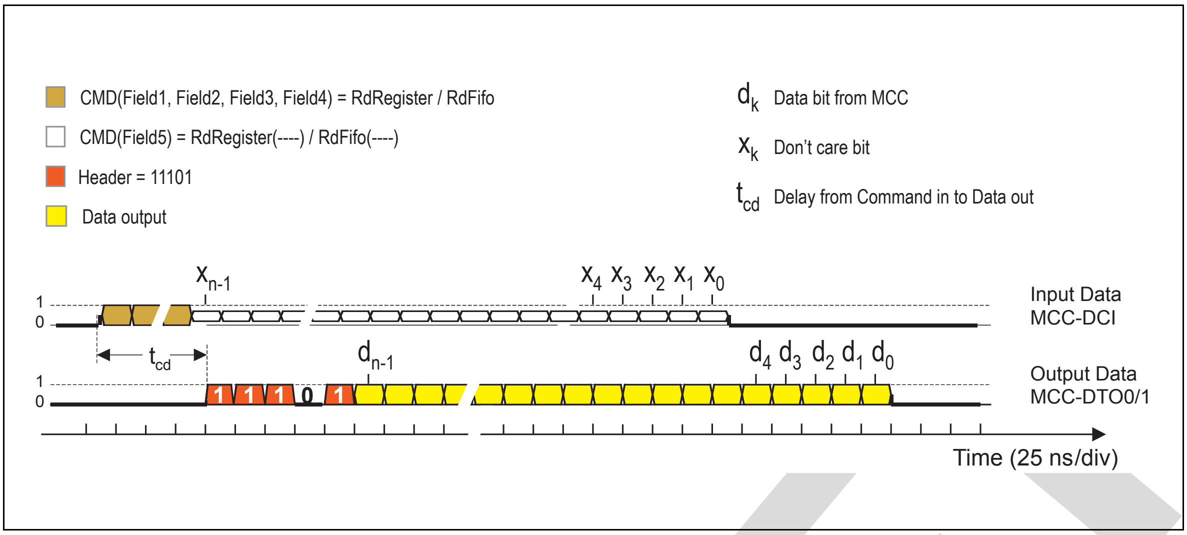
CMD(Field1, Field2, Field3, Field4) = RdRegister / RdFifo
CMD(Field5) = RdRegister(----) / RdFifo(----)
Header = 11101
Data output
d_k Data bit from MCC
x_k Don't care bit
t_{cd} Delay from Command in to Data out
x_{n-1}
x_4 x_3 x_2 x_1 x_0
1
0
Input Data
MCC-DCI
t_{cd}
d_{n-1}
d_4 d_3 d_2 d_1 d_0
1
0
1 1 1 0 1
Output Data
MCC-DTO0/1
Time (25 ns/div)

Figure 1-1 *MCC* to *ROD* data format in case of RdRegister and RdFifo commands.

1.1.1 Example of MCC configuration

Lets now assume that one wants to configure a module equipped with 16 working, and configured, *FE* chips, selecting a 160 Mbit / s output data transfer with ToT information and no consecutive Triggers.

The correct sequence to perform this operation is:

1. GlobalResetMCC: Resets the *MCC*.
2. WrRegister FEEN `0xffff`: Enables all 16 *FE* chips.
3. RdRegister FEEN: Checks that we wrote the correct value. This command returns the header followed by `0xffff` on both DTO-0 and DTO-1 lines.
4. WrRegister CSR `0x0013`: Enables ToT (`0010`) and sets the desired output speed (`0003`).
5. RdRegister CSR: Checks that we wrote the correct value. This command returns the header followed by `0x0013` on both DTO-0 and DTO-1 lines.
6. EnDataTake: Sets the *MCC* in RunMode.

We do not need to set the LV1 register as by default the *MCC* only sends one LV1 signal to all *FE*'s in response to a Trigger command.

1.2 MCC to FE Configuration Data Format

The interconnect topology between the *MCC* and the 16 *FE* chips in a module is a star topology which uses unidirectional serial links.

The *MCC* to *FE* protocol used to write and read *FE* chip configuration data uses three dedicated CMOS lines that are shared between all 16 *FE* chips hosted on a module.

The *FE* configuration commands are logically subdivided in three parts. The first part of the command is an *MCC* Slow command followed by a data part. This part is subdivided in a *FE* address plus control word and data word that are dealt with inside the *FE* chips.

To upload data to the *FE*'s, the *MCC* provides data (MCC-DAO) together with a clock to latch them (MCM-CCK) on the positive edge. The MCC-LD signal is used to distinguish, in the bit stream from the MCC-DAO pin, the address plus control words from the subsequent data words. The CCK is generated by the *MCC* only when serial data present at the DAO output must be latched in the *FE*. This clock has a lower frequency (5 MHz) than the XCK to reduce the timing requirements on the internal registers which can be loaded or read back using this protocol.

FE chips are selected for parameter read / write by serial geographical addressing.

Since some of the internal registers in the FE chip are organized as long chains of shift registers their readout will be destructive. Furthermore, no partial data loading is possible within each memory structure. These structures can be loaded with new data values while the old ones are shifted out and read back into the MCC through the FE-DO lines.

Figure 1-2 shows an example of a RdFrontEnd configuration command in which an internal *FE* register is loaded. As can be seen, in response to a serial command composed by the bits which identify the commend to the *MCC*, followed by the command and data that contains and the contents

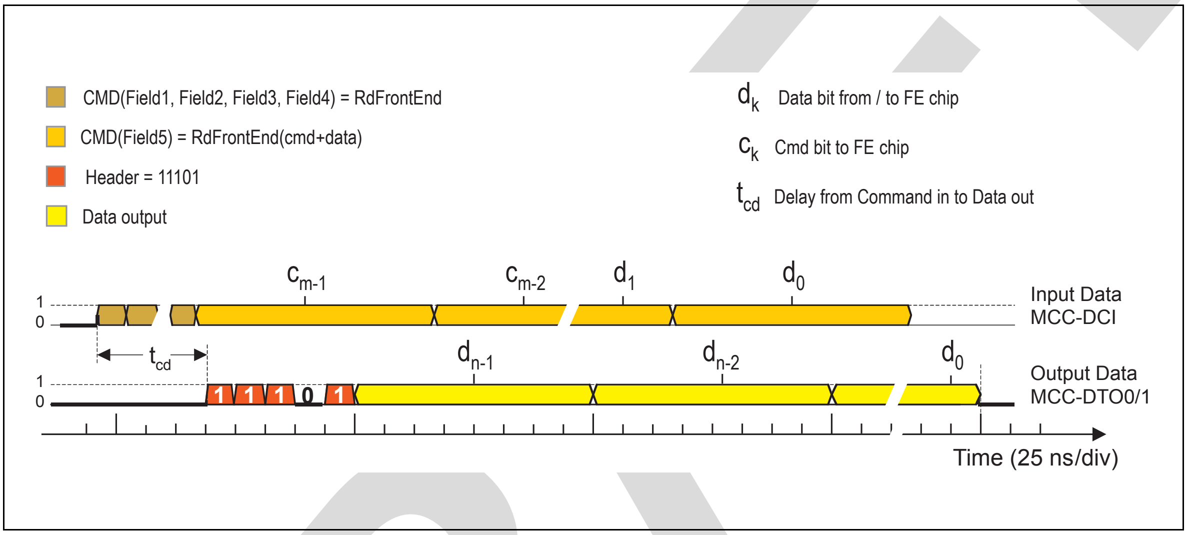

Figure 1-2 *MCC* to *ROD* data format in case of RdFrontEnd command.

Before transmitting the data train, the geographical address together with the command are shifted in (see Figure 1-3). Geographical address and command are transferred when LD is low. Then LD is raised and the transfer of the data starts. Data transmission is terminated by the falling edge of LD and the absence of CCK. CCK is generated by the *MCC* dividing its input clock (CK) by a factor 8 (i.e. CCK = 5 MHz). CCK is only generated when a bit carrying address, command or data information is present at MCC-DAO pin. The waveforms used by this protocol are shown in Figure 1-3.

In this figure we show only the *FE* part of the serial command issued to the *MCC*.

Figure 1-4 shows the waveforms of the signals in response to a *FE* configuration command depending on the settings of the CNT register. CNT<15:3> sets the length of the Data part of the *FE* command in CCK units, while CNT<2:0> x 8 sets the length of the Command part of the *FE* command. As LD is used to distinguish between Command and Data part we have to ensure that it is always generated.

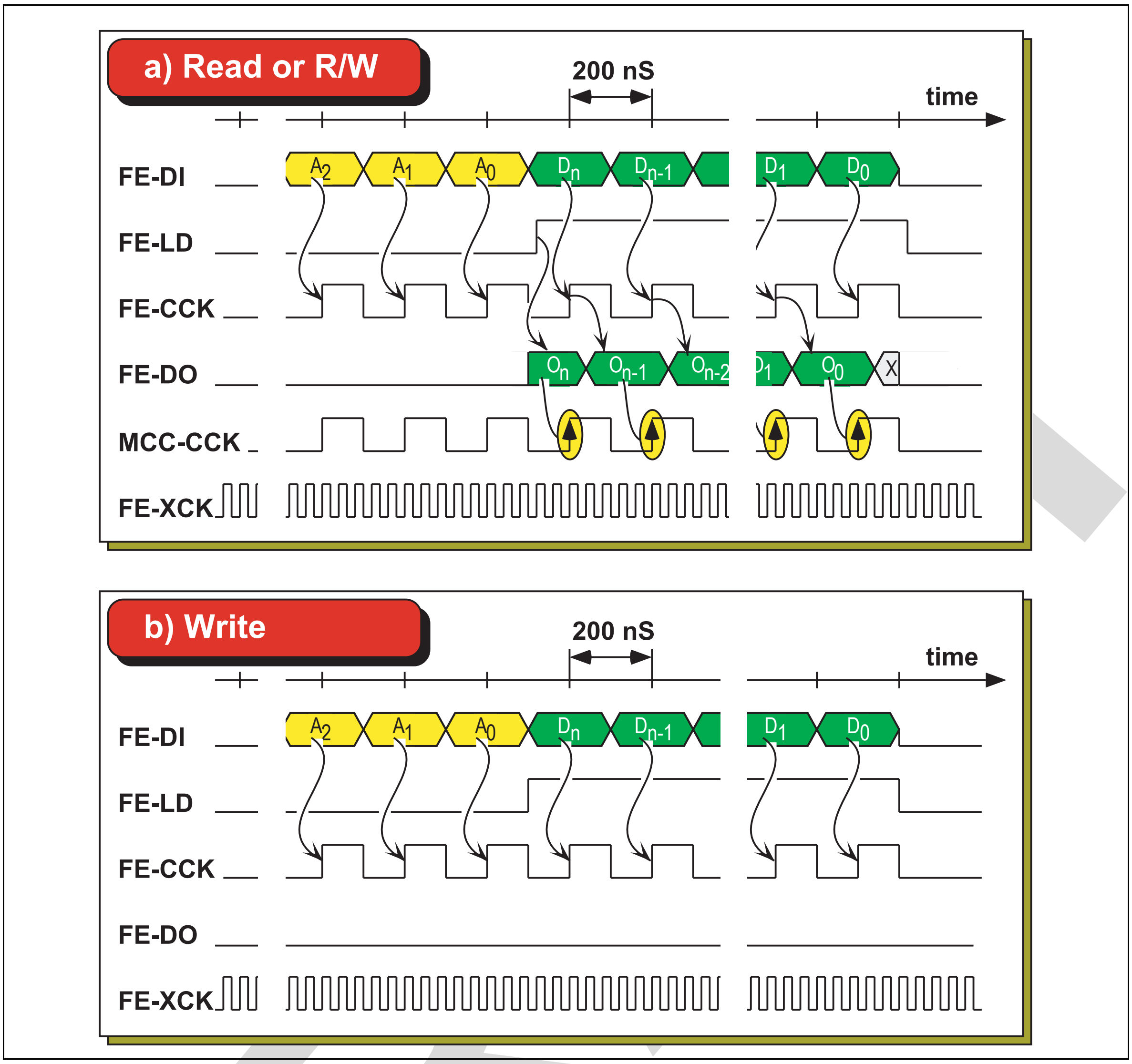

Figure 1-3 Protocol between *MCC* and *FE* chips to load initial configuration. Read/Write operations can be executed at the same time if new values are loaded into the *FE* while present values are readout (a). On low-to-high transition of the FE-LD signal only *FE* chips which recognize the transmitted address operate on data.

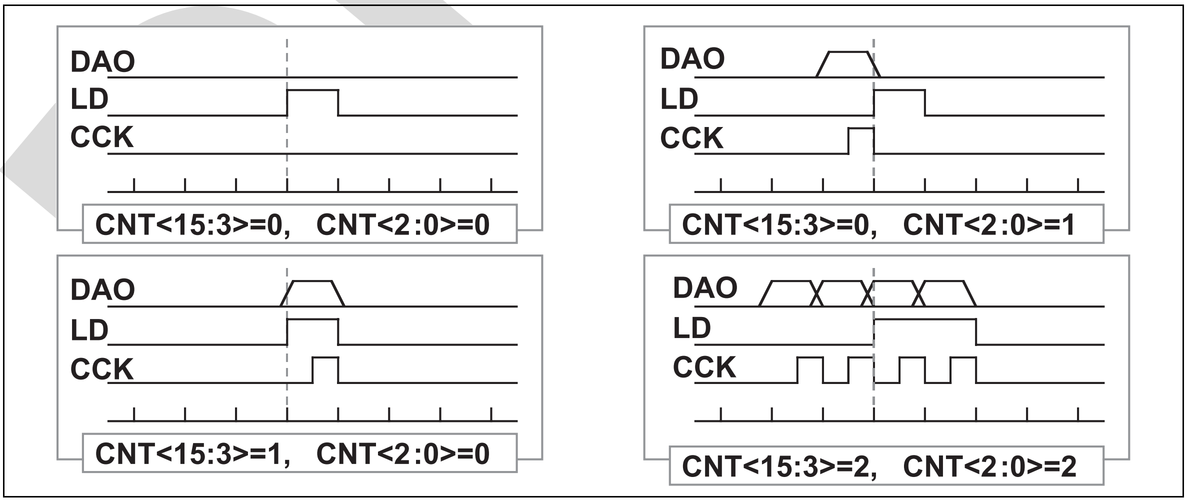
DAO
LD
CCK
CNT<15:3>=0,   CNT<2:0>=0
DAO
LD
CCK
CNT<15:3>=0,   CNT<2:0>=1
DAO
LD
CCK
CNT<15:3>=1,   CNT<2:0>=0
DAO
LD
CCK
CNT<15:3>=2,   CNT<2:0>=2

Figure 1-4 Waveforms of the signals to configure *FE* chips for different settings of CNT register.

1.3 FE to MCC Event Format

When the *MCC* is in RunMode, in response to a Trigger command, data is transmitted from the *FE* chips to the *MCC* via 16 dedicated LVDS lines (MCC-DTI<15:0>).

The *FE* chip encodes and transmits hits from an event using fixed length packets. Each packet contains a 1-bit header, followed by a 4-bit LV1, an 8-bit row number, a 5-bit column number, and an optional 8-bit ToT value. The encoding uses a non return to zero coding synchronous with the 40 MHz clock coming from the *MCC* (MCC-XCK). The *FE* encodes EoE (End-of-Event) and Warning messages, in case of hit overflow, using invalid, and therefore not used, row numbers. The Warning message is transmitted at the end of the event and replaces the EoE message, which is suppressed in such cases.

Table 1-1 summarizes the format for data hits and messages transmitted from *FE* chips to the *MCC*, while Figure 1-5 shows the transmission format in the time domain..

Table 1-1 Packet format for event read-out used in transmission from *FE* to *MCC*.

	Header (Bin)	**LV1# (Hex)**	**Row# (Hex)**	**Column# (Dec)**	**ToT (1)**
Data hit	1	0÷F	00 ÷ DF	0 ÷ 23	0 ÷ FF
EoE/WNG#n	1	0÷F	En	xx	xx
EoE	1	0÷F	Fn	xx	xx

Note:

(1) Time over Threshold (ToT) if option is selected, missing field if option is deselected

(2) x = don't care

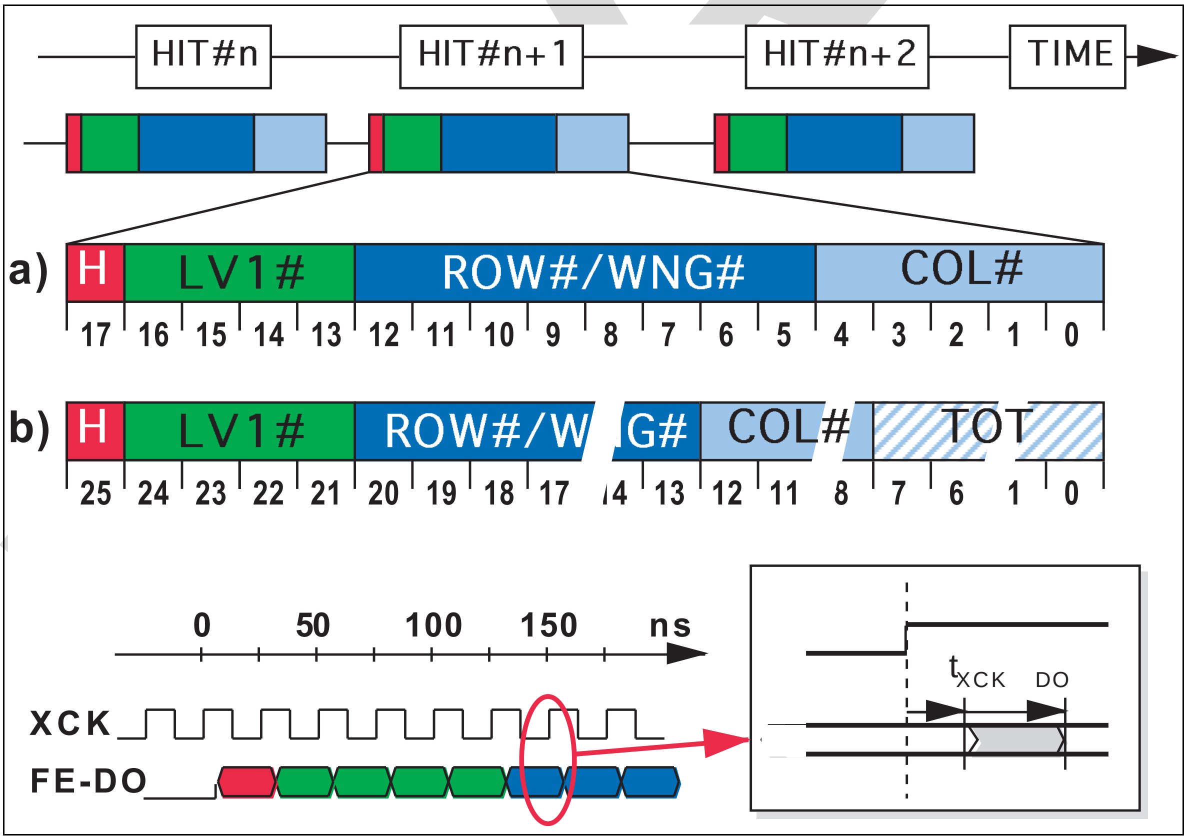

Figure 1-5 Data format and timing for event transmission from *FE* to *MCC*. Hit frame length is 18-bit long if Time over Threshold (ToT) is not provided (*a*), while it is 26-bit long if ToT is generated (*b*).

1.4 MCC to ROD Event Format

Figure 3-6 is a graphic example of event encoding from the *MCC* output.

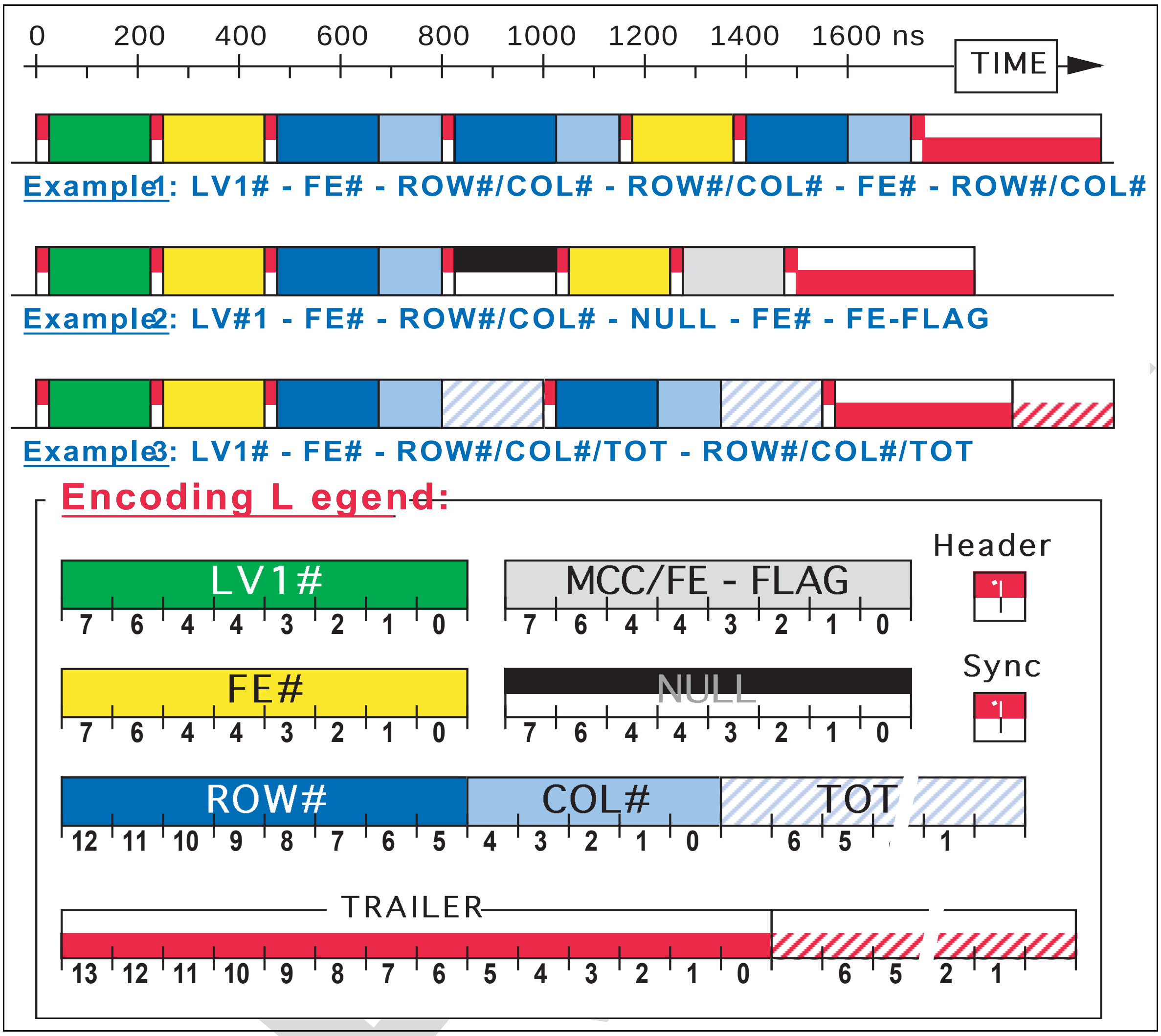
0 200 400 600 800 1000 1200 1400 1600 ns
TIME
Example1: LV1# - FE# - ROW#/COL# - ROW#/COL# - FE# - ROW#/COL#
Example2: LV#1 - FE# - ROW#/COL# - NULL - FE# - FE-FLAG
Example3: LV1# - FE# - ROW#/COL#/TOT - ROW#/COL#/TOT
Encoding L egend:
LV1#
7 6 4 4 3 2 1 0
MCC/FE - FLAG
7 6 4 4 3 2 1 0
Header
FE#
7 6 4 4 3 2 1 0
NULL
7 6 4 4 3 2 1 0
Sync
ROW#
12 11 10 9 8 7 6 5
COL#
4 3 2 1 0
TOT
6 5 1
TRAILER
13 12 11 10 9 8 7 6 5 4 3 2 1 0
6 5 2 1

Figure 1-6 Event Data format at the *MCC* output. The example 3 illustrates the case of ToT encoding.

The format of the event generated by the *MCC* has been defined considering that:

- Events are ordered by LV1 arrival time.

  Event building is done by grouping all hits belonging to the same LV1 number. The first event which has been triggered is the first to be sent out. Every event is entirely transmitted before the next event is considered for transmission: no event interleaving is allowed.

- Data are sorted and grouped by *FE* chips.

  The event builder sorts pixel hits by *FE* chip order. This permits data compression by “clustering” for either the case of a non-uniform hit distribution (jets) or a large pixel occupancy (B-layer) since every hit does not need to extend the address information to include the *FE* chip number with its row and column address.

- Event length is not known until the whole event is transmitted.

  The event builder does not know until the event is completely built up what the total number of hits is, or what the number of hits per *FE* chip is. This makes the event builder simpler, but forbids the usage of forward word counters in the event frame. Event data fields must be recognized by their content.

- Data must be compressed due to bandwidth limit.

  Since we need to keep the number of transmission cables as small as possible, and since a rather large amount of data is transmitted from the *MCC*, event coding which compresses data is specially important.

- Recovery from transmission errors.

  Any transmission error must be recovered by the next event data. The chosen mechanism is to use a Trailer whose value can never occur in the data. Since data fields are allowed to take any value we decided to add a Sync bit (bit set to '1') after every data field and a Trailer containing a fixed number of zeroes.

- Null data (wait).

  In the coding we have included the possibility of informing the receiver that the event is not finished yet in case the *MCC* cannot supply new data. This is optional for the *MCC*, but the design of the *ROD* must include this option for generality.

An event frame at the output of the MCC is an ordered stream of data fields separated by synchronization bits.

The syntax for the event data structure is as follow:

```
<Event>
   ::= <Header> L1ID <S> BCID <MccFlag>? <FrontEnd>* <Trailer>
<Header>
   ::= 11101
<S>
   ::= 1
<MccFlag>
   ::= <Sync> MCC-FLAG
<Sync>
   ::= <S>
   ||= <<S> NULL>+
<FrontEnd>
   ::= <Sync> MCC-FE# <Hit>+ <FeFlag>?
   ||= <Sync> MCC-FE# <Hit>* <FeFlag>
<Hit>
   ::= <Sync> ROW# COL#
   ||= <Sync> ROW# COL# TOT                    (if opt. ToT selected)
<FeFlag>
   ::= <Sync> FE-FLAG
<Trailer>
   ::= <S> 00 0000 0000 0000
   ||= <S> 00 0000 0000 0000 0000 0000         (if opt. ToT selected)
```

The following items summarize the format of the formal syntax description:

<Name> name in lower case is a syntax construct defined by other syntax constructs or by a bit field.

NAME in upper case is a bit field. It's definition is stated in Table 1-2.

<Name>? is an optional field, 0 or 1 items occurrence.

<Name>* is 0, 1 or more items.

<Name>+ is 1 or more items.

::= gives a syntax definition to an item.

||= introduces an alternative syntax definition.

The values of the keywords which appear in the event syntax are in Table 1-2. In Figure 1-7 and Figure 1-8 are shown the flow diagrams of the encoding/decoding algorithms that follow the grammar rules for the event data format.

⚠ The 8 bit of the LV1Id field are splitted in two distinct fields: LV1Id<7:4> contains the number of skipped events, while LV1Id<3:0> contain a 4 bit LV1Id number. Both numbers are generated by the TTC (see XXX).

Table 1-2 Value for the keywords in the event syntax.

Keyword	Low Value	High Value	Description
BCID	0000 0000	1111 1111	Bunch Crossing ID (0÷255)
COL#	0 0000	1 0111	Column number
FE-FLAG	1 1110 FFFF MMMM		Error / Warning: F=*FE*, M=*MCC*
LVID	0000 0000	1111 1111	Skipped Level 1 (0÷15) and Level 1Trigger ID (0÷15)
MCC-FE#	1110 0000	1110 1111	*FE* number in the module from 0 to 15
MCC-FLAG	0000 0000	1101 1111	Error / Warning or Message from the *MCC*
NULL	1111 1111		Null data
ROW#	0000 0000	1101 1111	Row number from 0 to 224
TOT	0000 0000	1111 1111	Time over Threshold (if opt. ToT is selected)

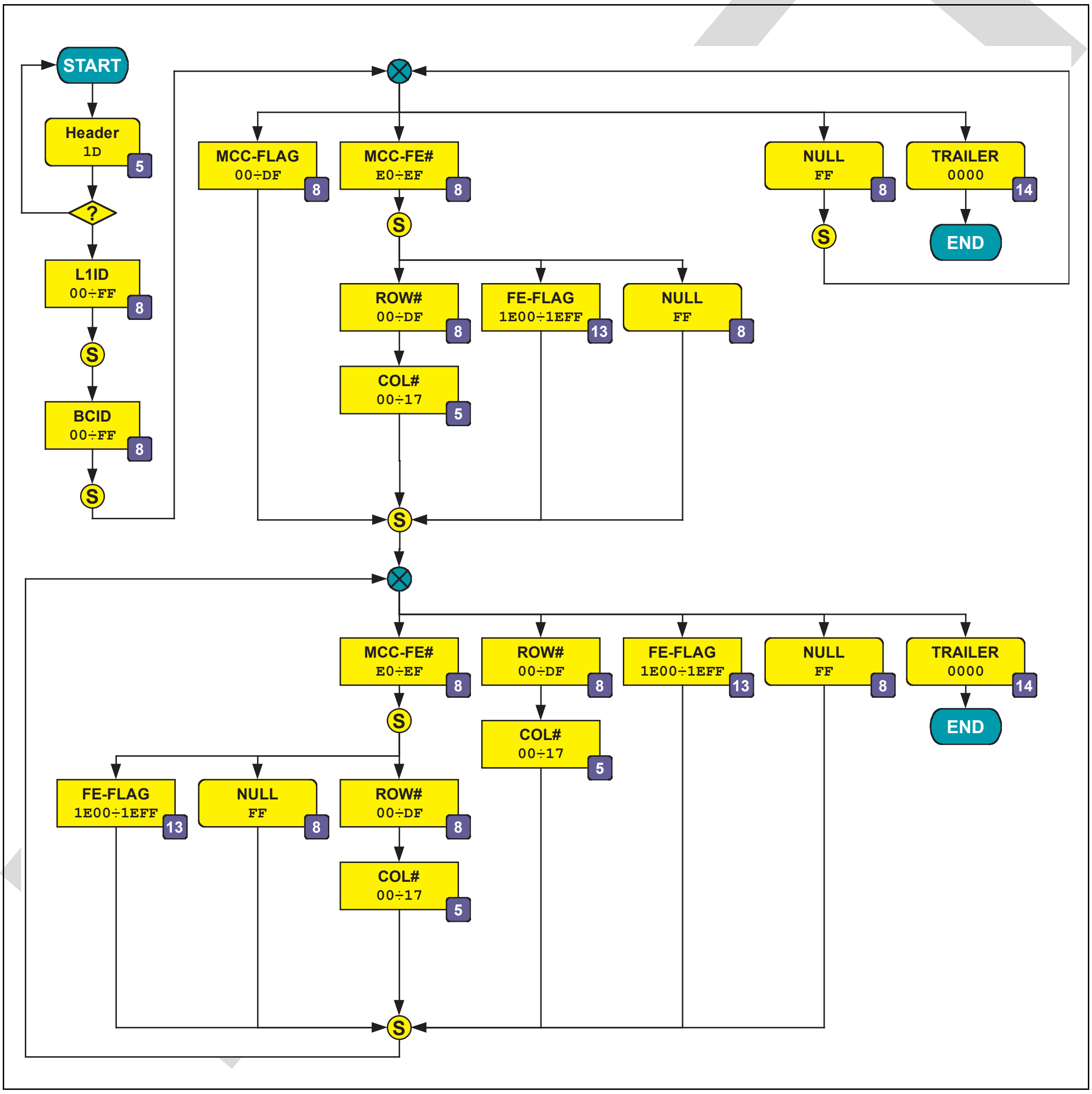

Figure 1-7 Encoding/decoding algorithm. No time over threshold case.

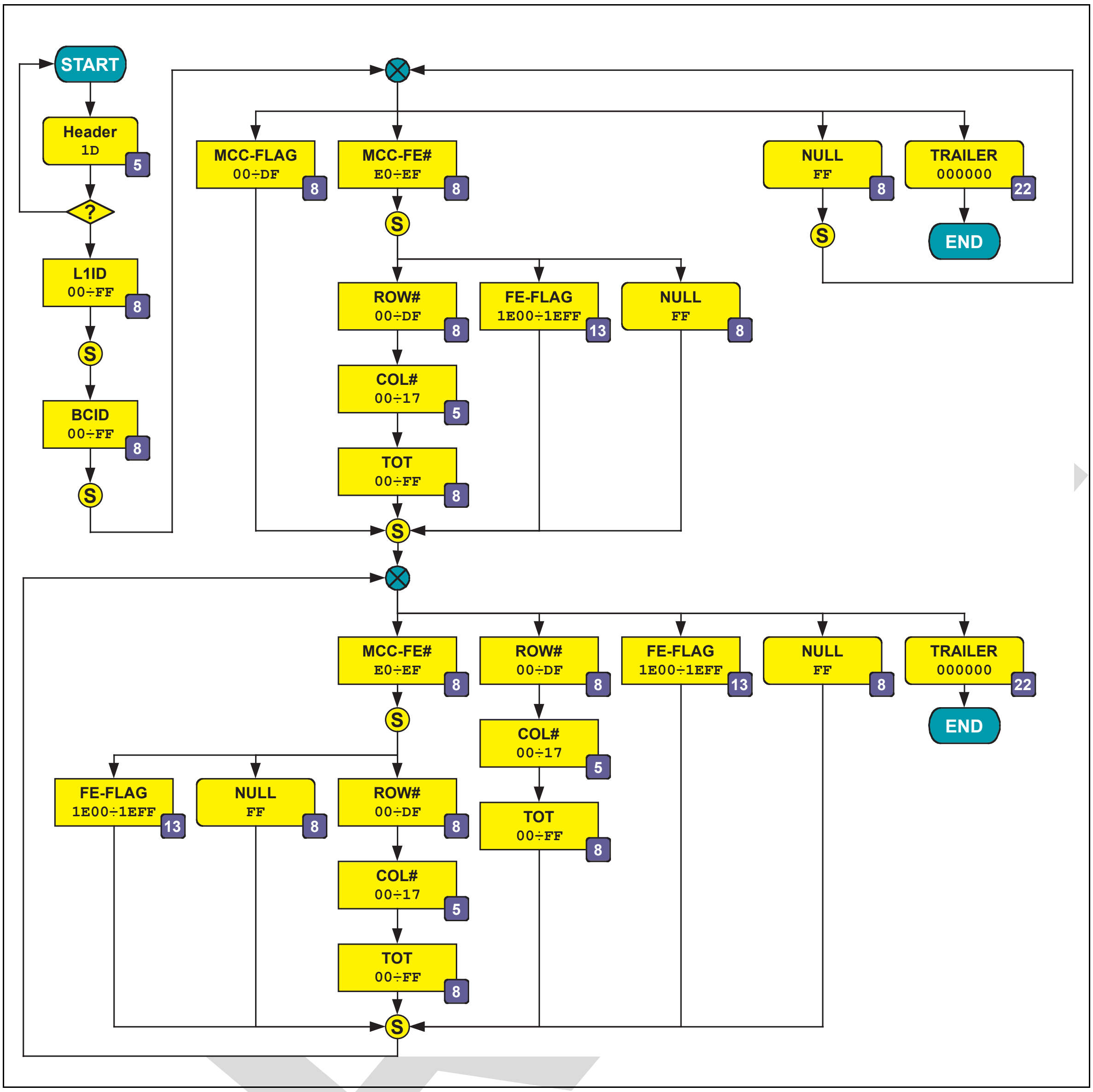

Figure 1-8 Encoding/decoding algorithm. Time over threshold case.

1.5 MCC to ROD Physical Layer Protocol

The *MCC* has two pins which can be used to drive the *VDC* chip which in turn converts the LVDS signal to the suitable currents to drive the opto fibre going to the *ROD*. The *MCC* has therefore different ways to transmit data to the ROD through the opto-links. Each link can be used as a single 40 or 80 Mb/s link. In addition the two links can operate as a single or a dual link. This allows for a peak bandwidth going from 40 Mb/s to 160 Mb/s.

The four operation modes are listed in the Table 1-3 and are graphically represented in Figure 1-9. The 4 link modes are selected by the two bits in CSR. In case of mode 40 x 1 and 80 x 1 the two links always transmit the same information and either one can be used. In case of dual link usage (40 x 2 and 80 x 2) the first bit of every event record always starts on DTO-1 output, while the last bit may be either on DTO-1 or on DTO-0.

Table 1-3 CSR bits setting for different link operation modes.

Mode	CSR<7:6>	Event Data	Event BW	Config. Data	Config. BW
40 x 1	0 0	2 links @ 40 Mb/s	40 Mb/s	2 links @ 40 Mb/s	40 Mb/s
40 x 2	0 1	2 links @ 40 Mb/s	80 Mb/s	2 links @ 40 Mb/s	40 Mb/s
80 x 1	1 0	2 links @ 80 Mb/s	80 Mb/s	2 links @ 40 Mb/s	40 Mb/s
80 x 2	1 1	2 links @ 80 Mb/s	160 Mb/s	2 links @ 40 Mb/s	40 Mb/s

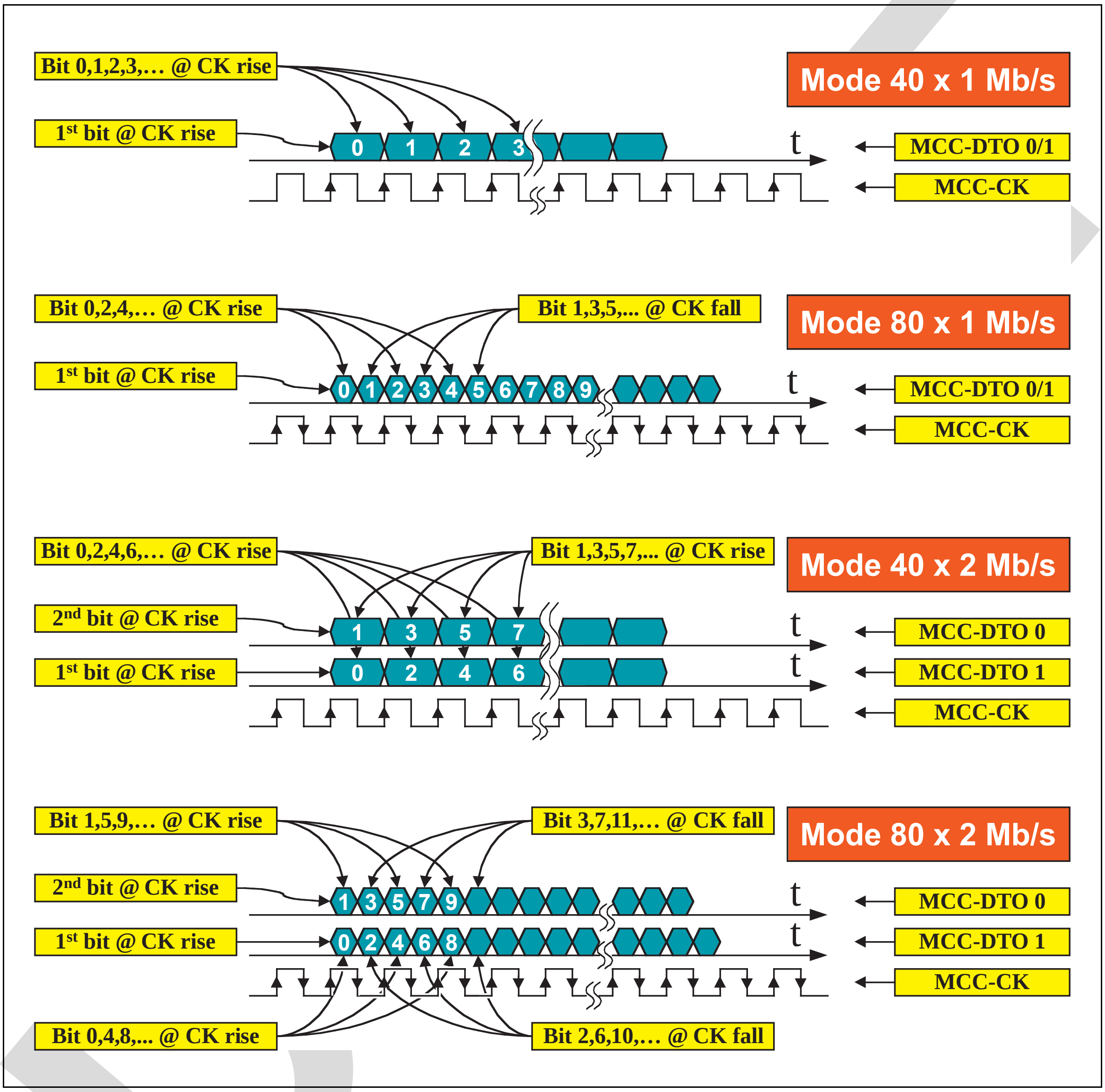

Figure 1-9 Data transmission on the two *MCC* outputs to the *ROD* for different link operation modes.

All configuration data, instead, is always transmitted at 40 Mb/s on both links independently from the link mode setting.